BURGE HUGHES WALSH

The Systems Engineering Tool Box

Dr Stuart Burge

"Give us the tools and we will finish the job"

Winston Churchill

Holistic Requirements Model (HRM)

What is it and what does it do?

The Holistic Requirements Model is derived from a systems approach to the classification of system requirements. It provides a consistent analysis framework that can be used when interpreting, clarifying and deducing system requirements from a set of customer/stakeholder requirements.

Applying systems thinking to the requirements of a system leads to the Holistic Requirements Model shown in Figure 1.

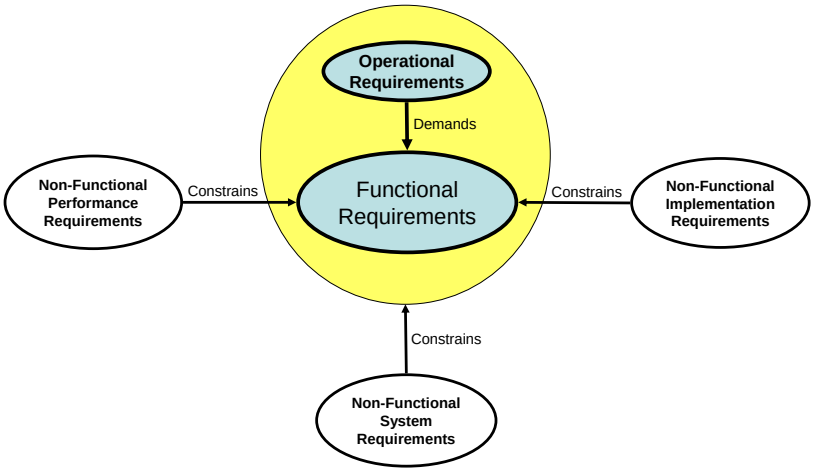

Figure 1: The Holistic Requirements Model[1]

The Holistic Requirements Model is so-called because it provides a complete and consistent model for classifying and organizing any set of requirements of a system. Furthermore, it is only truly understandable as a whole, and isolated consideration of the component requirement types is ephemeral.

[1] This requirements model has its origins in the work performed by BAe (Now BAE SYSTEMS) in defining a software/systems tool called CORE (Controlled Requirements Expression.

© Stuart Burge 2006

The model comprises three basic requirements types:

- Operational Requirements
- Functional Requirement
- Non-Functional Requirements.

With a further sub-classification of the Non-Functional Requirements set

- Non-Functional Performance Requirements
- Non-Functional System Requirements
- Non-Functional Implementations Requirements.

Why do it?

There are many ways in which a set of requirements can be classified and indeed examining requirements from different viewpoints can often be enlightening. It is, however, extremely useful to have at least one consistent approach to the categorisation of requirements since this will allow for a consistent and repeatable interpretation of the incomplete, in correct and ambiguous requirements provided by customers and other stakeholders. The HRM provides such, if not the only, approach since it is firmly rooted in systems theory.

Where and when to use it?

The Holistic Requirements Model is perhaps more of a framework than a tool since it provides a ubiquitous structure for many systems engineering tools[2]. It is a model of requirements that should be understood by every engineer and form part of their unconscious competence. Indeed, it provides a "way" of looking at requirements that is beyond conventional interpretation.

It can be used in a "stand-alone" fashion or via other tools like Systemic Textual Analysis to analyse and interpret any set of requirements whatever their origin.

How to do it?

The Concept

Systems theory states that all systems have a purpose and context. Furthermore, the system purpose can often be logical decomposed into a number of necessary sub-purposes, sub-sub-purposes etc. These two Systems theory facts lead to the definition of the Operational and Functional requirements of a system as:

[2] These tools include: Systemic Textual Analysis, Viewpoint Analysis, Functional Modelling, Quality Function Deployment to name but a few.

© Stuart Burge 2006

Operational Requirements define the major purpose of a system (i.e. what it fundamentally does; its capability) together with those key-overarching constraints (that define the context of the system). For example:

System	Operational Requirement
Toaster	To toast bread-based products evenly and safely
Dish Washer	To clean eating and cooking utensils efficiently and without damage.
Civil Aircraft	To transfer passengers and their baggage from one point to another efficiently and safely.
Hotel Reception	To register guests accurately and quickly in a professional manner.

The Operational Requirement is a succinct clear and unambiguous statement as to what the system fundamentally does together with the key constraints. It cannot be emphasized enough how important the Operational Requirement of a system is – all systems will have them – but they may not be written down. Experience has shown that customers rarely specify Operational Requirements because they believe it is obvious. They are not always obvious and it is important to expend effort in developing Operational Requirements that all parties are happy with. There are three reasons for this:

1. Operational Requirements provide precise direction for the system development team. Without an Operational Requirement individual team members will develop their own internal version and, although these may be similar, they will still be 'different' and those differences will obviate any collective focus.
2. The Operational Requirements will demand certain system functionality that forms the basis of the Functional Requirements. In other words, for given Operational Requirement, the laws of physics and logic dictate that a number of lower-level functions (jobs if you like) have to be performed.
3. The Operational Requirement defines the prime system and thereby anchors the remainder of the model.

Functional Requirements specify the functions of the system. What the system has to do in order to achieve the Operational Requirement. They capture and define the sub-purposes of the overall system purpose (given in the operational requirements). Since Functional Requirements specify the functions of the system that are best described as a *verb-noun* phrase that defines an action on an object. For example, some of the functions of a civil aircraft are:

- Navigate Aircraft
- Manoeuvre Aircraft
- Generate Lift
- Store Passengers
- Control Passenger Storage Environment
- Communicate with other aircraft and ATC
- Etc

© Stuart Burge 2006

There are several points to note about categorising and defining Functional Requirements:

- A Functional Requirement defines what has to be done – not how it is done or how well it is done.
- The function in a Functional Requirement is best defined as a verb-noun phrase:
  - A requirement can have a verb but not be a function! For example the requirement; "*easy to use*" has a verb but this is not a function. The statement "*the system does easy to use*" does not make sense, but "the system has to be easy to use" does make sense a property.
  - The best verbs are active imperative regular verbs as opposed to passive irregular verbs. Therefore, having a verb in a requirement is a necessary but not sufficient condition for a Functional Requirement.
- Functions can often be viewed as transformers of inputs into outputs.
- When identifying Functional Requirements the 'system of interest' should be clearly defined.
- When defining Functional Requirements performance qualifiers should be avoided. For example:
  - Toast bread quickly
  - Load bread easily

  specify the function (what has to be done) and how well it is to be done. How well a function has to be accomplished is the purpose of Non-Functional Performance Requirements.
- Functional Requirements should be implementation independent. For example, the use of the expression "store passengers" is deliberate to avoid the use of "seat passengers" which is a solution
- There are often several possible choices for the verb to describe what the system has to do. For example:
  - Store Passengers
  - Accommodate Passengers
  - Locate Passengers
  - Restrain Passengers

  All provide a similar description of the function. Wherever possible the verb should be selected to give a neutral description that does infer a solution.
- Beware of pseudo functions. These are verb noun combinations that do not define an action on an "object". For example "provide reliability" does not define an action on an OBJECT but demands or infers the presence of a property or the achievement of a certain level of performance. Pseudo functions can be identified since they employ an abstract noun – something that cannot be touched like style, reliability, safety etc. True functions use concrete nouns.

© Stuart Burge 2006

Appendix A provides a table of useful and also not so useful verbs for defining functional requirements.

Like Operation Requirements, customers and stakeholders rarely specify Functional Requirements them. This is not surprising since most customers and stakeholders are interested in how well something is done, rather than what is done. In practice, this means that customers and stakeholders will imply Functional Requirements through either performance expectations or through providing solutions expressed as constraint based non-functional requirements.

Non-Functional Requirements are constraints on the system that define how well something is to be done or how it is to be done which fall into three categories:

- **Non-Functional Performance Requirements**
- **Non-Functional System Requirements**
- **Non-Functional Implementation Requirements**

These categories are derived from consideration of the essential relationships with the Operational and Functional Requirements and ensure the completeness and consistency of the requirements model. It is these essential relationships between the functionally-based requirements and constraint-based requirements that make the Holistic Requirements Model so useful. Indeed, it is exploiting the relationships that the assignment of a set of source requirements into the various HRM categories that permits the deduction of missing requirements.

- **Non-Functional Performance Requirements** are associated with corresponding Functional Requirements and define how well a particular function has to perform – they are the constraints on that function. For example:

Function	Non-Functional Performance Requirement
Generate Heat	Heat density > 5.75kw/m^2 Max Q < 10 seconds
Navigate Aircraft	Accuracy < ±1 mile in 3,000 Precision standard deviation < 2.4 miles
Heat Water	Accuracy < ±1^{o}C of set value Time to set value < 10 minutes

Non-Functional Requirements are quantitative comprising a measure, property or parameter together with a numerical value.

© Stuart Burge 2006

- **Non-Functional System Requirements** define the constraints that affect the whole or a significant proportion of the system and include:

  - Physical Attributes
    - System Style
    - System Size
    - System Weight

  - The '-ilities'
    - System Reliability
    - System Maintainability
    - System Interoperability
    - System Availability

  - System Performance
    - Cost
    - Speed
    - Manoeuvrability.

  It is important to note that there are two categories of Performance Requirements - those that are associated with a specific function (Non-Functional Performance Requirements) and those that are associated with the whole system (Non-Functional System Requirements). It is important (but sometimes difficult) to distinguish between them. In the early stages of system development, particularly if the system is unprecedented, it may not be clear if a particular Performance Requirements is at the Functional or System level. If this is the case, it should be categorized wherever suitable but subject to constant review.

- **Non-Functional Implementation Requirements** define how a system is to be built in terms of specific technology. These may be specific requirements from the customer about a solution that they require or they may be legislative requirements.

System	Function	Non-Functional Implementation Requirement
Toaster	Receive Power	UK domestic 13 amp plug to BS 1363
Dishwasher	Remove Waste	Electric pump
Civil Aircraft	Communicate with other aircraft and ATC	Phillips A/C 1267 VHF radio

Non-Functional Implementation requirements have two sources; they can be *justifiable* or *lazy*!

© Stuart Burge 2006

- Justifiable: where the requirement is reasonable typically because relates to a design decision made at the higher system level.
- Lazy: where the stakeholder has resorted to a solution based requirement because they do not know how to express it otherwise.

The Holistic Requirements Model

These requirement types allow for the construction of the Holistic Requirements Model (HRM) shown in Figure 1. This model is driven by the Operational Requirement and contains the Functional Requirements at its heart. It is through the suitable implementation of the Functional Requirement that the Operational Requirement(s) is delivered. The Non-Functional Requirements describe the expectation levels of the customer/stakeholder and constrain the functionality.

At this point, it is very important to understand that the HRM is applicable to any system at any level. This makes the HRM very powerful in that it is universally applicable. But this power comes at price, which is that care must be exercised when transferring requirements between system levels. Systems theory states that a system comprises sub-systems, and that a system is a sub-system of a bigger system. Relating this concept to the HRM indicates that system functions are sub-systems particularly at high levels of generality. Consider a domestic washing machine which can be considered as a system and which can be defined by a HRM. One of the functions of a domestic washing machine is to “drain water”. The “drain water” function can be treated as a sub-system of the washing machine system. If the drain water function is considered in isolation it is a system in its own right and can be defined its own HRM. Clearly the two HRMs are different but must be related. The relationship is two-fold:

1. The appropriate Functional Requirement of the system becomes the purpose element of the Operational Requirement of the sub-system.
2. The Non-Functional Performance Requirements of the function of the system become Non-Functional System Requirements of the sub-system, some of which (the critical ones) complete the Operational Requirement of the sub-system.

This can be illustrated with the domestic washing machine example. Figure 2 shows a partially completed HRM for the washing machine.

© Stuart Burge 2006

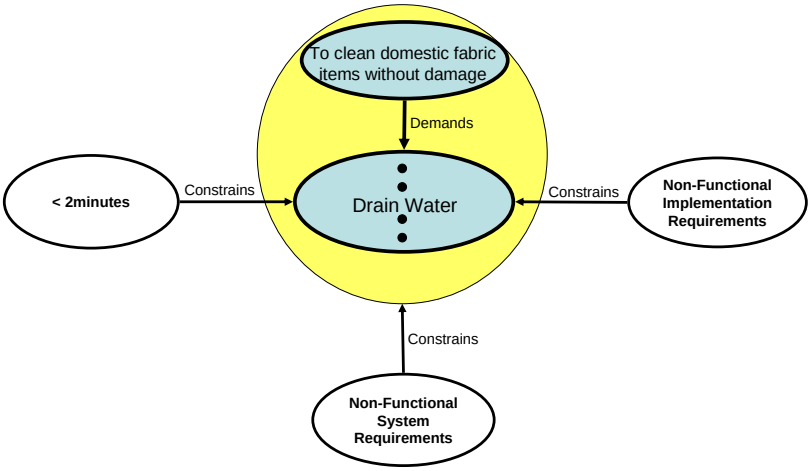

Figure 2: Partial HRM for the Washing Machine

Figure 2 shows the washing machine system having an Operational Requirement that demands 'Drain Water' functionality which is constrained by the Non-Functional Performance Requirement of < 2 minutes. Consider now that the "Drain Water" is selected as a sub-system of the washing machine system. This sub-system is of course a "lower level system" for which it is possible and desirable to apply the HRM. Figure 3 shows a partially complete HRM for the 'Drain Water' sub-system.

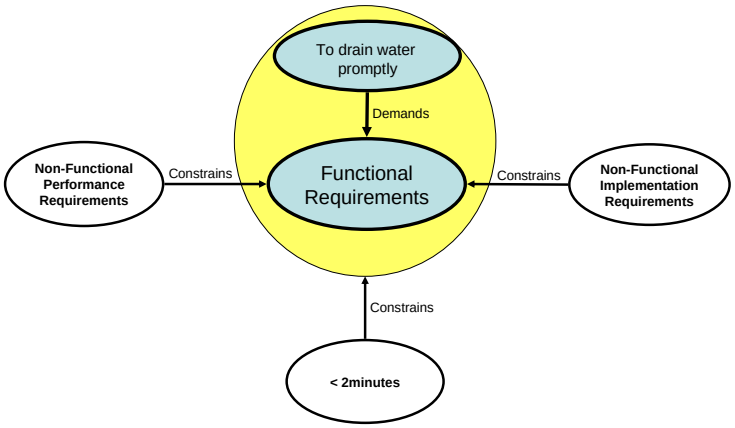

Figure 3: Partial HRM for the Drain Water Sub-System (Function)

Figure 3 shows how the washing machine Functional Requirement to 'Drain Water' becomes part of the Operational Requirement of the HRM for the 'Drain Water' sub-system. Furthermore, Figure 3 shows how the Non-Functional Performance Requirement of draining in < 2 minutes is a Non-Functional System Requirement for the 'Drain Water' sub-system. This performance is also included in the Operational Requirement of the 'Drain Water' sub-system.

Figure 4 summarises these relationships between system levels and the Holistic Requirements Model.

© Stuart Burge 2006

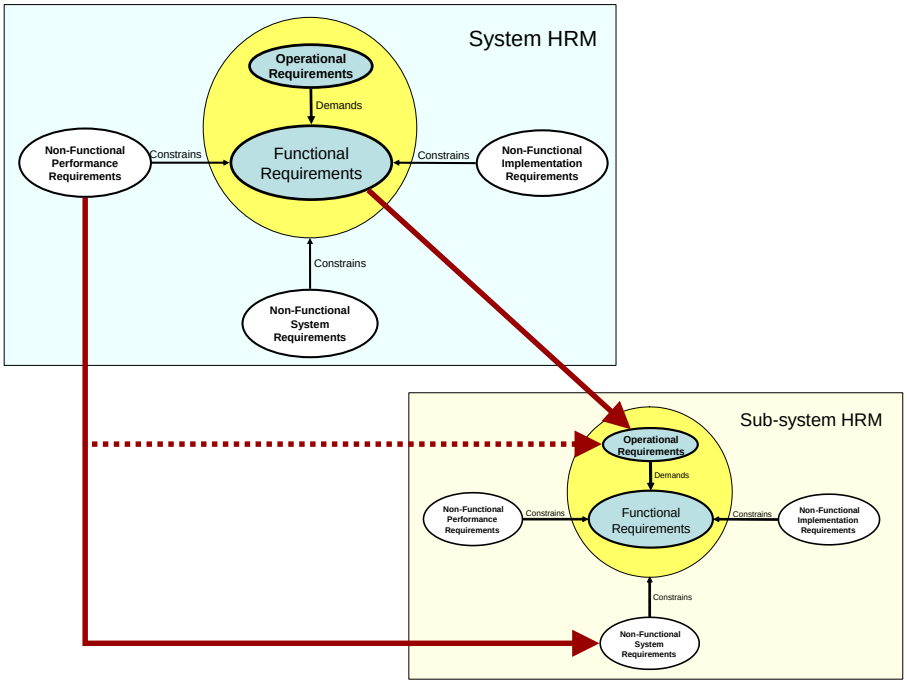

Figure 4: Relationship between System Level and Associated HRMs

How to use it?

The Holistic Requirements Model is not a tool as such, but a reference model that is used by Systems Engineering tools such as Systemic Textual Analysis and Viewpoint Analysis. It also aligns itself to other tools that pre-date its creation, such as Functional Modelling and Quality Function Deployment. Indeed, both these tools are enriched by the alignment of the HRM since they provide a structural framework that makes their application somewhat easier and considerably more powerful.

In essence the relationships between the requirements types in the HRM can be exploited to uncover and deduce unspoken customer and stakeholder requirements. For example, consider some typical requirements for a toaster as shown in figure 5[3].

[3] These are an actual set of requirements collected from a training course where a small team of delegates were given the question: "What would be your requirement for a domestic toaster?"

© Stuart Burge 2006

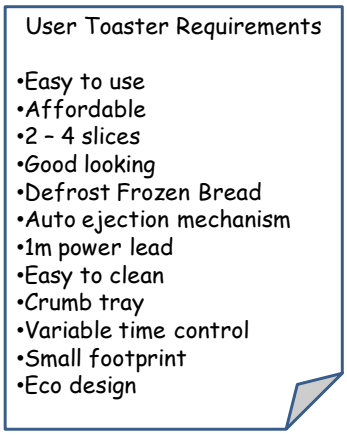

Figure 5: a set of user requirements for a domestic toaster

These requirements can be analysed using the HRM by taking each in turn and categorising it. The categorisation can then be used to deduce missing requirements since:

- A Non-Functional Implementation (NFI) Requirement is the solution to a Functional (F) Requirement. Therefore given a NFI requirement it is possible to deduce the corresponding function it solves.
- A Non-Functional Performance (NFP) Requirement defines how well a particular system function has to perform. Therefore given a NFP requirements it is possible to deduce the corresponding function to which it applies.
- A Functional (F) requirement is an ephemeral requirement – it only defines the function – the job – that needs to be done. To provide completeness requires the associated Non-Functional Performance requirements. Therefore if Functional requirements are provided without any associated NFP requirements these will have to be determined.
- Every system will have a set of underlying functionality (generic functionality) necessary for it to achieve its Operation Requirements. Given an Operation Requirement it should be possible using logic and experience to deduce the necessary functionality (and thereby functional requirements) to deliver the purpose.

These demands and inferences are shown diagrammatically in Figure 6.

© Stuart Burge 2006

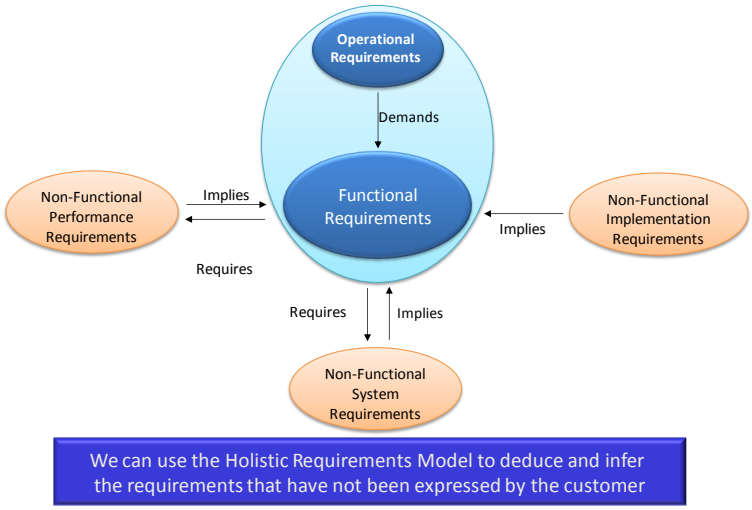

Figure 6: Deduction and inference using the HRM

In order to use the relationships given in figure 6 requires the correct categorisation of a set of requirements. The flow chart shown in Figure 7 can help in this process.

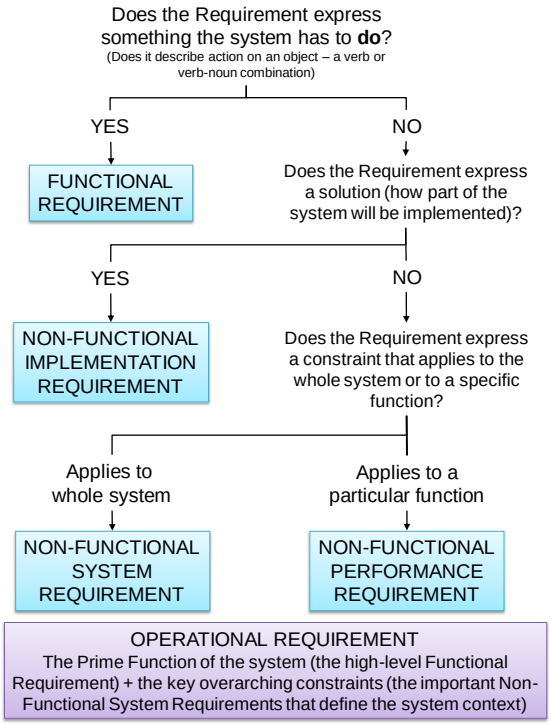

Figure 7: HRM requirements categorisation flow chart

© Stuart Burge 2006

If the flow chart is used on the toaster requirements the following categorisations emerge:

Requirement	HRM Category	Reason
Easy to use	NFS	System does not "do" *easy to use* – it has to **be** *easy to use*. The whole system has to be *easy to use*
Affordable	NFS	As above
2 – 4 slices	NFS or NFP	Could affect whole system or just one function – does imply LOAD BREAD function however
Good looking	NFS	As *easy to use*
Defrost Frozen bread	F	Something the system will have to do – however, note this is not a generic function found in every toaster
Auto ejection mechanism	NFI	The system does do *auto ejection mechanism*, therefore not a function. It does, however, **have** a *auto ejection mechanism* –a thing hence NFI. Note that behind every NFI is a Function that can be deduced
1m power lead	NFI	Power lead infers electrical power supply and hence a solution. Does imply Functional requirement of SUPPLY POWER or POWER TOASTER
Easy to clean	NFS	As *easy to use*
Crumb tray	NFI	A clear solution to the functional requirement COLLECT CRUMBS or MANAGE CRUMBS
Variable time control	NFI	This requirement suggests that the toasting level is achieved by varying the toasting time. The same result could be achieved by fixing the time and varying the heat level – i.e. a solution to the function of CONTROL TOASTING
Small footprint	NFS	As *easy to use*
Eco design	NFS	As *easy to use*

What Goes Wrong: The limitations of HRM

Sources requirements document contains requirements for multiple systems: Many source requirements documents often contain requirements for the prime system that is to be developed and requirements for other associated systems. A typical situation is the inclusion of requirements for the project or programmes to deliver the prime system. It is important when using the HRM to be clear on what system a particular requirement belongs to and categorize it relative to that system.

Endless debates about performance requirements: Performance requirements can potentially be classified using the HRM as either Non-functional Performance or Non-functional System. During the early stage of system development it can be difficult to determine whether a particular performance requirement is Non-functional System or Non-functional Performance. This is typically because it is either not adequately defined or due to a lack of understanding at this early point about system

© Stuart Burge 2006

functionality (the Functional Requirements). In such instances the starting point to try categorising such requirements as Non-functional Performance and testing whether this categorization leads to the derivation of any new functionality. If it does then the categorization was correct. If a single function cannot be identified then it is better to categorise the performance requirement as Non-Functional System. In some instances the performance requirement implies many functional requirements. In this instance the Functional Requirements should be captured but the performance requirement categorised as a Non-functional System Requirement.

© Stuart Burge 2006

Appendix A: Useful verbs in defining Functions and Functional Requirements

Absorb	Contain	Finish	Limit	Plan	Stamp
Accelerate	Control	Fix	Learn	Position	Start
“Access”	Convert	Flash	Load	Power	Steer
Accommodate	Convey	Flow	Locate	Press	Stop
Act	Cook	Fly	Lock	Prevent	Store
Activate	Cover	Force	Log	Print	Submit
Actuate	Create	Form	Look	Produce	Supply
Adapt	Cut		Lubricate	Prognose	Support
Add		Generate		Protect	
Agitate	Dampen	Glide	Maintain	Pump	Tap
Adjust	Decide	Grasp	Manage		Target
Advise	Define	Grind	Manoeuvre	Receive	Teach
Alert	Deflect	Grip	Manufacture	Rectify	Tend
Align	Deliver	Group	“Market”	Reduce	Throw
Analyse	Demonstrate	Grow	Measure	Regulate	Transfer
Apply	Depress	Guide	Metre	Release	Transmit
Approve	Design		Meet	Relocate	Transport
Assemble	Detect	Heat	Mill	Remove	Trim
Assign	Determine	Hold	Mix	Repair	
Assure	Diagnose	House	Modify	Reserve	Unite
Attach	Differentiate		Modulate	Resist	Unify
Attenuate	Direct	Identify	Move	Restrain	Unload
Attract	Display	Illuminate		Retain	Update
	Dispense	Impede	Navigate	Rinse	
Balance	Dispose	Improve	Negate	Rotate	Validate
Blend	Dispatch	Imbibe	Neutralise	Route	Verify
Bore	Distribute	Increase	Notify		Visualise
Burn	Drain	Indicate		Scrap	Vibrate
Buy	Drill	Induce	Observe	Seal	Vaporise
Bury		Inform	Obtain	Secure	
	Eject	Initiate	Order	Select	Warn
Calculate	Eliminate	“Input”	Organize	Sell	Wash
Carry	Emit	Inspect	Orient	Send	Weigh
Check	Enclose	Instruct	“Output”	Sense	Watch
Circulate	Enter	Insulate	Overhaul	Shear	Weld
Clean	Expel	Integrate	Override	Shift	Wet
Collate	Extend	Invoke		Sign	Wipe
Communicate	Extract	Isolate	Paint	Ship	Wire
Conceal			Pay	Sound	Wring
Conduct	Fasten	Jettison	Perform	Space	
Conserve	Feel	Join	Permit	Spin	Zip
Constrict	Fill	Jump	Pivot	Squeeze	Zoom

© Stuart Burge 2006

Guidance

Look for VERBS that don't imply a solution:

Deliver Toast (good)
Eject Toast (bad because it suggests a "pop-up" mechanism)

Do explore different verbs. There is always a choice. But start by capturing your first thoughts (usually they will reflect the current solution) and then explore alternatives Use the **imperative verbs** (command verbs) given in the table wherever possible . The verbs in quotes strictly speaking are not verbs in UK English they are nouns but common usage has turned them into verbs!

To test whether a word is a verb try "to" before of add "ing". For example "to safety" or "safetying" do not make sense and therefore safety is not a verb.

Not so useful verbs in defining functions that often lead to the creation of "Pseudo Functions"

Achieve
Allow
Appoint
Conform
Cope
Enhance
Encourage
Exceed
Facilitate
Improve
Meet
Provide
Promote
Satisfy

These verbs are often used to turn non-functional requirements into Pseudo Functions. For example Provide Reliability, Improve safety, Facilitate availability. The point to note here is that a Functional Requirement specifies what the system has to do in terms of:

an ACTION on an OBJECT

Pseudo functions like "provide reliability" do not define an action on an OBJECT but demand or infer the presence of a property or the achievement of a certain level of performance. Avoiding pseudo functions also requires consideration of the noun. The NOUN defines the object the verb acts on. It should be a **concrete noun** something that be touched, seen, tasted, smelt or heard.

Nouns like reliability, style, safety are not concrete nouns by but **abstract nouns.** sThese are things you cannot touch, see, hear, smell or taste (but can sometime measure) like reliability, safety, style, etc.

© Stuart Burge 2006